SOLID principles

- **S - Single Responsibility Principle (SRP)**

  Definition: A class should have only **one reason to change**, meaning it should do **only one job**.

  - For example, a `Report` class should only handle report data, not save it to a file—that should be another class's job.

- **O - Open/Closed Principle (OCP)**

  Definition: Software entities (classes, modules, functions, etc.) should be **open for extension but closed for modification**.

  - You should be able to **add new functionality** without changing existing code.
  - For example, if we have a general function for calculating the area of a circle, it can't accept other shapes; every time we have a new shape, we have to change the function (add a new function). So it's better to make a general (parent) class for all the shapes; others could be inherited by this, so their methods and functions can be inherited with them.

- **L - Liskov Substitution Principle (LSP)**

  Definition: Subclasses should be **substitutable for their base classes** without altering the correctness of the program.

  - Derived classes must behave in a way that does not break the parent class's expectations.
  - Agar `Bird` class mai `fly()` hai, to `Penguin` uska child na ho, kyun ke Penguin urh nahi sakta.
    Agar kiya to code toot jayega — **LSP toot gaya**.

- **I - Interface Segregation Principle (ISP)**

  Definition: Clients should **not be forced to depend** on methods they do not use.

  - Break large interfaces into smaller and more specific ones.
  - Bari interface banane ki bajaye, choti-choti interfaces banao taake har class sirf apna kaam kare.

🔹 D - Dependency Inversion Principle (DIP)

Definition: High-level modules should **not depend on low-level modules**. Both should depend on abstractions.

- Details should depend on abstractions, not the other way around.

🔚 In Short:

❌ Galat Tarika	✅ Sahi Tarika (DIP)
Computer → WiredKeyboard	Computer → IKeyboard ← Wired/Wireless